הטכניון – מכון טכנולוגי לישראל

הפקולטה להנדסת חשמל

Laboratory of Networked Software Systems

בית חכם עם רשת חברתית

אביב תשע"ח

מגיש: אלכסיי חרומוב מנחה: אורן קלינסקי

תוכן עניינים

1. מבוא

חברות הטכנולוגיה הגדולות יודעות עלינו הכל, אז למה לא לנצל אותן להציג לנו מראש את מה שאנחנו אוהבים?
ניקח דוגמה: אדם קם בבוקר, רוצה לראות מה חדש בעולם החדשות, הספורט או הסדרות האהובות עליו, הוא כבר עשה "לייק" לעמודים האהובים עליו בפייסבוק, נשאר רק שהמחשב יציג בשבילו את העדכונים החדשים מתוכם!
כאן הבית החכם נכנס לפעולה: נעמיד מחשב raspberry pi3 עם מצלמה ומסך מחוברים אליו, המצלמה תקלוט ותזהה את האדם העומד מולה, המסך יציג את התוכן האהוב עליו מאחד עמודי פייסבוק.
כעת נניח שהאדם קרא את העדכון הכי חדיש, והוא רוצה לעבור לפוסט הבא, או שהוא דווקא רוצה להחליף עמוד, למה "להתאמץ" בלחיצת כפתורי מקלדת \ עכבר, אם אפשר פשוט לעשות תנועות ידיים באוויר?
נכון! המערכת תזהה את תנועות הידיים ותפעל לפיהם.

2. אתגרים בפיתוח אפליקציה חכמה

לפני פיתוח האפליקציה עמדו לפנינו אתגרים רבים. ראשית התממשקות עם SDKs שונים (יפורטו בהמשך), עמידה בדרישות אבטחה של פייסבוק, זיהוי מדויק של תנועות ידיים בעזרת מצלמה ולבסוף שימוש במחשב raspberry pi3 לתפעול האפליקציה.
במהלך הפרוייקט נעשה שינוי במערכת זיהוי המשתמש, התחלנו להשתמש בAWS, והגיע אתגר נוסף: הענן מגדיר מספר מקסימלי של תמונות חינמיות לזיהוי, ואסור לחרוג ממספר זה.

- האתגר הכי גדול בפיתוח התגלה דווקא לאחר ניסוח הפרוייקט:
מתוך ויקיפדיה:

קיימברידג' אנליטיקה)באנגלית (**Cambridge Analytica**: הייתה חברה פרטית בריטית ששילבה כריית נתונים, מסחר בנתונים, ניתוח מידע ואסטרטגיה תקשורתית לצורך השפעה על תהליכי בחירות במדינות ברחבי העולם[...] בשנת 2016הייתה החברה מעורבת במסע הבחירות לנשיאות ארצות הברית של דונלד טראמפ ובקמפיין הברקזיט ליציאת בריטניה מהאיחוד האירופי .תפקודה של קיימברידג' אנליטיקה בקמפיינים הללו הוא נושא לחקירות פליליות מתמשכות בשתי המדינות, בעיקר בנוגע לשיטות בהן היא נוקטת לצורך מיקוד בוחרים[...] ב 17-במרץ, 2018 דיווחו העיתונים" ניו יורק טיימס "ו"האובזרבר "שקיימברידג' אנליטיקה עשתה שימוש עסקי במידע אישי מפייסבוק ,שמלכתחילה נאסף על ידי חוקר חיצוני למטרות אקדמיות. בתגובה, אסרה פייסבוק על קיימברידג' אנליטיקה לפרסם בפייסבוק.

האירועים הנ"ל גרמו לפייסבוק לבצע שינוי משמעותי באבטחה. החל מאפריל 2018 לא ניתן לתת הרשאות publish_actions לאפליקציות – הרשאה שהיה בה צורך להעלאת תמונה לקיר של המשתמש הראשי בעזרת API שנקרא FB-RECOG.
נזכיר שמטרות הפרוייקט הוגדרו במרץ 2018.
הניסיון הבא היה להעלות את התמונות לעמוד (page) של פייסבוק, במקום לקיר של משתמש, אך גם פתרון זה נפל! הפעם משום שפייסבוק לא מאפשרים בעזרת graph api לתייג תמונות בעמוד, גם הפעם לצרכי אבטחה ופרטיות.
הפתרון שלבסוף פתר את האתגר בוצע בעזרת Amazon Web Services, ומערכת הRekognition שלה, נפרט עליו עוד בהמשך.

- אתגר שצץ לאורך הפרוייקט ומתקשר לAPI של פייסבוק, הינו אופן הצגת המידע. תחילה הצגנו דף של פייסבוק, וניתן היה לגלול בין דפים בעזרת תנועות ידיים לצדדים. אך הסתבר שלא ניתן לגלול את העמוד למעלה ולמטה עם תנועות ידיים (או בכלל מהקוד), פייסבוק לא חושפים את החלקים הפנימיים של הPLUGIN שלהם לשינוי ע"י קוד.
לאחר מחשבה הוחלט לבצע "פתרון מקורי", במקום להציג את דף הפייסבוק, נציג פוסט אחד בלבד בכל פעם, ועם תנועות ידיים מעלה ומטה, נגלול בין פוסטים של אותו עמוד.
פתרון זה עבד ללא קושי, מלבד העובדה שהיה צורך לבצע פארסינג גם לעמוד פייסבוק אהוב (בנוסף לפארסינג לכלל העמודים האהובים של המשתמש) כדי למצוא לינקים לפוסטים.

- אתגר נוסף שעמד לפנינו היה שליפת תמונה מתוך streaming של מצלמה והעברתה לשרת לצורך זיהוי. פתרון שלבסוף לא נלקח משום שלא היה יעיל: לעשות THREAD בצד שרת של המצלמה, ופעם בחצי שנייה לעצור אותו, לעשות שליפת תמונה ואז להמשיך את הSTREAMING. הפתרון כמובן לא טוב, אין סיבה לעצור את הזרמת המצלמה בשביל לקחת תמונה, זה עלול לגרום לתקיעות ולפעולות מיותרות של התוכנה.
הפתרון שכן נלקח לבסוף הוא שימוש במצלמה מצד הלקוח, שירות הSTREMING מבוצע ישירות בצד לקוח ע"י JAVASCRIPT, פעם בחצי שנייה הסקריפט ייקח SCREENSHOT מתוך המצלמה, בלי לעצור את הזרמת הנתונים למסך, ויישלח את הלינק הזמני של התמונה לצד שרת. השרת ייגש ללינק הזמני, ישמור את התמונה ולאחר מכן יעבד אותה – בין אם מדובר בזיהוי פנים ובין אם מדובר בזיהוי תנועות.

- אתגר אחר שעמד לפנינו היה זיהוי תנועות ידיים. על מנת לזהות את התנועות השתמשנו באלגוריתם הבא: לקחנו 2 תמונות עוקבות בהפרש של חצי שנייה ועשינו על שתיהן השוואה של תנועה. האלגוריתם עבד בצורה סבירה, אך לא מספיק מדויקת. עקב עובדה זו הוחלט להשתמש באלגוריתם מורכב יותר אשר ניתן למצוא בפרק "ארכיטקטורה ומימוש".

- הרצת התוכנה על על RASPBERYY PI3 היוותה אתגר נוסף. רוב המעברים מחלונות לRaspbian עברו בצורה חלקה, אך לא ספריית OPENCV, (עבור זיהוי תנועות ידיים). ספרייה זו איננה ניתנת להורדה למערכת הפעלה זו.

- כפי שצוין, במהלך הפרוייקט צץ אתגר נוסף: לגרום לפרוייקט להשאר חינמי.
AWS מאפשרים העלאת 5000 תמונות בחודש בלבד לצורך זיהוי בחינם, אם היינו מעלים תמונה כל חצי שנייה לבדיקה של הגעת משתמש חדש, היינו חורגים מהר מאוד מהמספר הנ"ל. לכן פותח אלגוריתם "קמצן" שמעלה תמונה לזיהוי רק במקרה הצורך. פירוט מורחב יותר בפרק "ארכיטקטורה ומימוש".

- אתגר מהירות התוכנה: כמו כל תוכנה, גם אפליקציה זו הייתה חייבת להיות בעל תגובה מהירה עבור המשתמש. ראספברי פאי איטי, גם האינטרנט לא תמיד מהיר (בייחוד בויפי), הניסיון הראשון היה לתת לדפים האהובים עם הפוסטים שלהם להטען ביחד לזיכרון ברגע שמזוהה משתשמש. היתרון של ניסיון זה היה מהירות תגובה אדירה בעת תזוזות ידיים, והחיסרון הוא שמרגע זיהוי משתמש, ועד להצגת דפים לקח כ40 שניות (וזה תחת הנחה שלוקחים מספר מועט של עמודים ופוסטים בכל אחד).
החסרון עלה על היתרון, ולבסוף הוחלט לטעון כל פוסט בנפרד כשהמשתמש עושה גלילה.

נקודה זו מתקשרת לבעיית עבודה על ראספברי פאי, מכיוון שעליו הטעינה אפילו איטית יותר. הפתרון שנלקח עובד מצוין ביחד עם הנקודה האחרונה (שתגיע מיד). אחרי זיהוי תנועה, נשלחת הודעה לצד לקוח שהתנועה אכן זוהתה, כעת צד הלקוח יבקש מהשרת את הפוסט הבא להצגה. אם הייתה גלילה לאחד הצדדים, יהיה על השרת לבקש מפייסבוק את העמוד האהוב הבא ולפרסר אותו, ואילו אם הייתה תנועה למעלב או למטה, אזי לשרת כבר יש את הלינק, והתגובה תהיה מיידית. כך פתרנו את בעיית המהירות תגובה.

- אתגר אחרון אשר עמד לפנינו לאורך הפרוייקט היה סנכרון של תקשורת אסינכרונית בין צד לקוח וצד שרת. לדוגמה: בזמן זיהוי בפנים אין צורך לבקש זיהוי תנועות ידיים, הקריאות עלולות אפילו להפריע אחת לשנייה!

הפתרון הוא כמובן לגרום לעבודה אסינכרונית, שלרוב יעילה יותר, לעבוד באופן סינכרוני, שלרוב איטי יותר אך יציב יותר. השתמשנו ב"מנעול" בצד לקוח, אשר עוצר שליחת תמונות מהמצלמה (וגם את העיבוד שלהן) עד שלא חזרה תשובה על תמונה קודמת.

3. SDKs & APIs

facebook-jssdk

רץ בצד לקוח ומתממשק עם javascript.
SDK שאנחנו מקבלים מפייסבוק, אחראי על יצירת חלון עם עמוד פייסבוק (או פוסט פייסבוק) באפליקציה שלנו, אשר מכיל את התוכן המבוקש, במקרה זה – אחד העמודים האהובים על המשתמש הנוכחי. נשים לב שהעמוד או הפוסט חייב להיות PUBLIC, מכיוון שאנחנו ניגשים אליו דרך המשתמש הראשי, שייתכן ולא עשה לו לייק.

AWS - Rekognition

ענן של אמאזון, בתוכו יש API של שירות Rekognition. השירות הינו מאוד חזק ומספק מגוון רחב של אפשרויות זיהוי פנים וניתוח תמונה. השימוש שלנו בAPI בוצע כך:

- בשימוש הראשון של משתמש, עליו להזין את השם משתמש הייחודי שלו בפייסבוק (לא השם כפי שמופיע בעמוד, אלא username), וללחוץ על כפתור "משתמש חדש".
- השם נשלח ביחד עם תמונת משתמש לAPI, ושם נשמר וקטור מידע על תווי הפנים של האדם.
- כעת לAPI יהיה אוסף וקטורים של תווי פנים של משתמשים רבים, וניתן בעזרת שליחת תמונה לAPI, לבקש זיהוי של האדם הנוכחי מול המצלמה.
- אם נמצאה התאמה מדויקת של מעל 90% בין האדם מול המצלמה ואדם באוסף, אפשר להציג את תוכן העמודים האהובים עליו בפייסבוק.

BEAUTIFULSOUP

ספרייה בפייתון אשר באה לידי שימוש בצד שרת. הספרייה אחראית על PARSING של עמוד HTML אותו היא מקבלת כקלט (URL). במקרה שלנו, אנחנו מזינים לאובייקט של BEAUTIFULSOUP כתובת של עמוד הלייקים של המשתמש הנוכחי, האובייקט ניגש לעמוד, ומפרסר אותו. באחריותנו לקחת את הקובץ XML שיצר הפרסר, ולמצוא בתוכו את האובייקט (CLASS) אשר מכיל בתוכו לינק לעמוד אהוב.
נמצא את כל העמודים האהובים של המשתמש, ונחזיר אותם כמילון של שם + לינק. לאחר מכן ניקח את הלינקים לעמודים האהובים, וגם אותם נשלח לאובייקט נוסף של BEAUTIFULLSOUP על מנת למשוך מהם לינקים לפוסטים בעמודים הנ"ל, ואת הפוסטים נוכל להציג באפליקציה.
הערה חשובה: התוכן שמוחזר מדף HTML מוחזר בתוך הערה, טרם פרסור הHTML, עלינו להוריד את סימני ההערה, על מנת שהBEAUTIFULSOUP יוכל למצוא את תגיות XML הדרושות.

FLASK

API בפייתון שאחראי על הרצת אתר במתודולוגיה הסטנדרטית של שרת-לקוח. במקרה שלנו השרת הינו מחשב וירטואלי שאנו מפעילים טרם הרצת האפליקציה.
הAPI הוא זה שמאפשר לנו ליצור תקשורת בין הלקוח והשרת, ע"י פנייה של הלקוח ללינקים מוגדרים מראש. לדוגמה פנייה של הלקוח לכתובת: "/my_link" תגרום לקריאה הבאה של הפונקצייה בצד שרת:

```
@app.route("/my_link", methods=['GET','POST'])
def getHandGesture():
...
```

ניתן להעביר מידע מלקוח לשרת בעזרת שליחת מילון data, במבנה ג'ייסוני ביחד עם הבקשה. השרת יקרא את המידע בעזרת גישה לארגומנטים, לדוגמה:

```
path = request.args.get('image_src')
```

התקשורת בין הצדדים היא בעזרת סטנדרט JSON (ראה נספח). מקור לFLASK:
http://flask.pocoo.org/docs/1.0/

OPENCV4 + NUMPY

ספריות של PYTHON אשר אחראיות על עיבוד תמונה, בעזרת ספריות אלו אנו מפענחים את תנועות הידיים של המשתמש. לאלגוריתם של הפענוח, ראה פרק "ארכיטקטורה ומימוש".

4. מבט על

האפליקציה עובדת בצורת תכן סטנדרטית של צד לקוח וצד שרת:

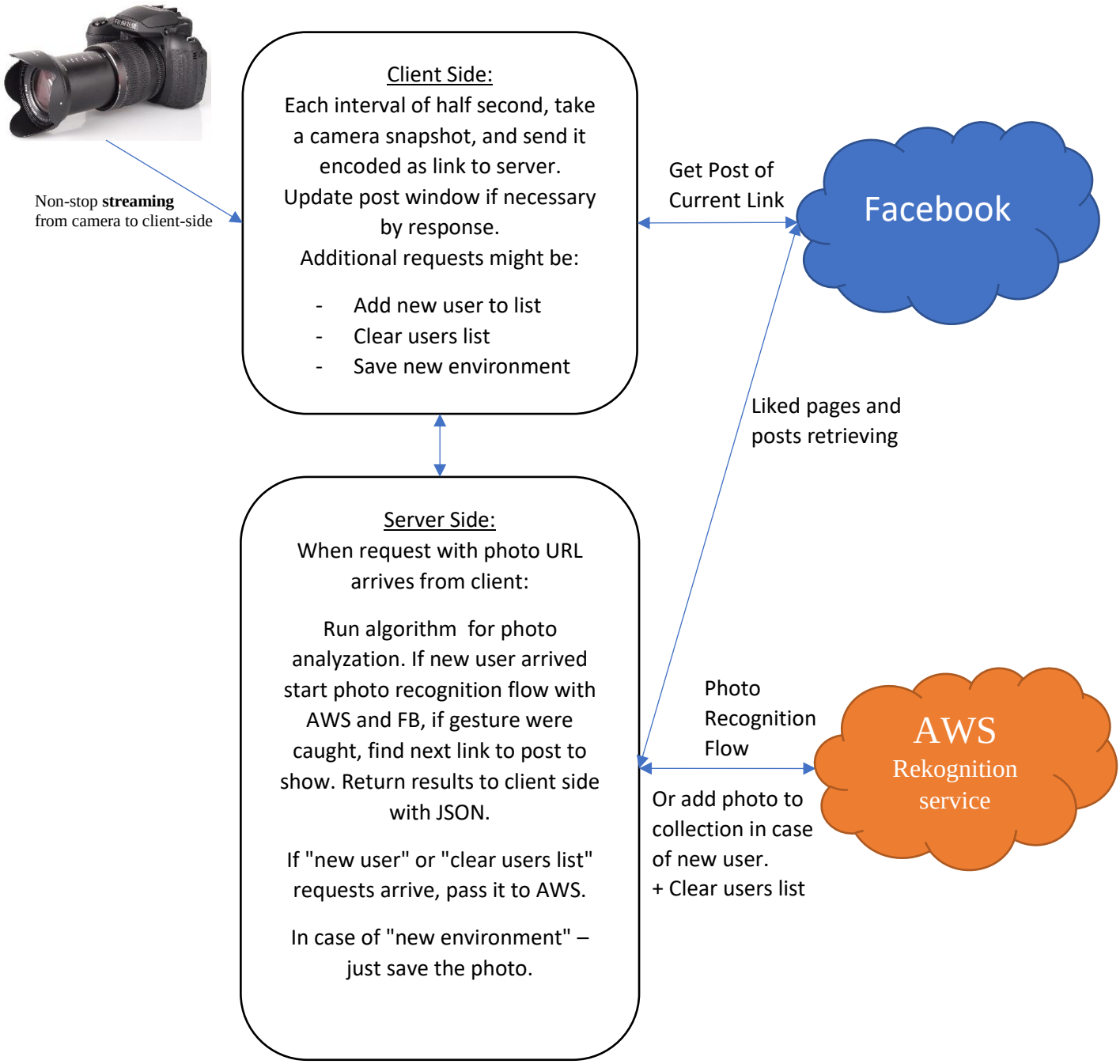
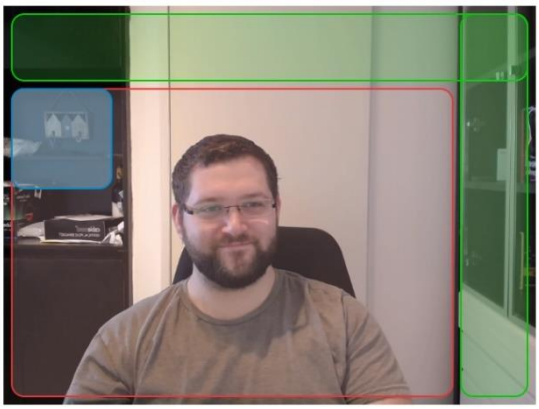
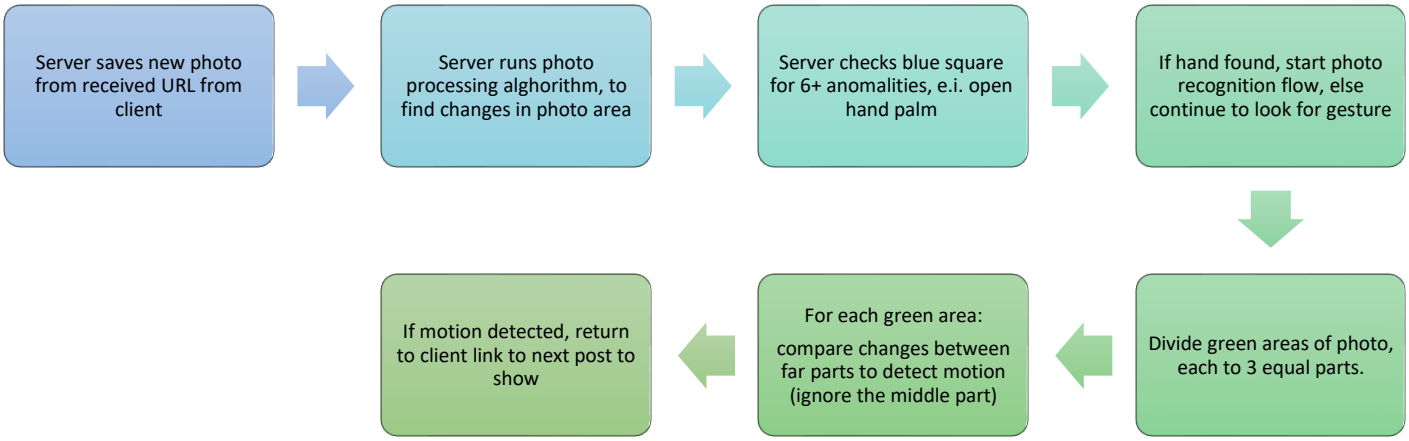


Photo Analyzation

על מנת לעבד באופן מדויק את התמונה נחלק אותה בצורה הבאה (החלוקה מופיע בצד לקוח על המצלמה, למען דיוק בתנועות):

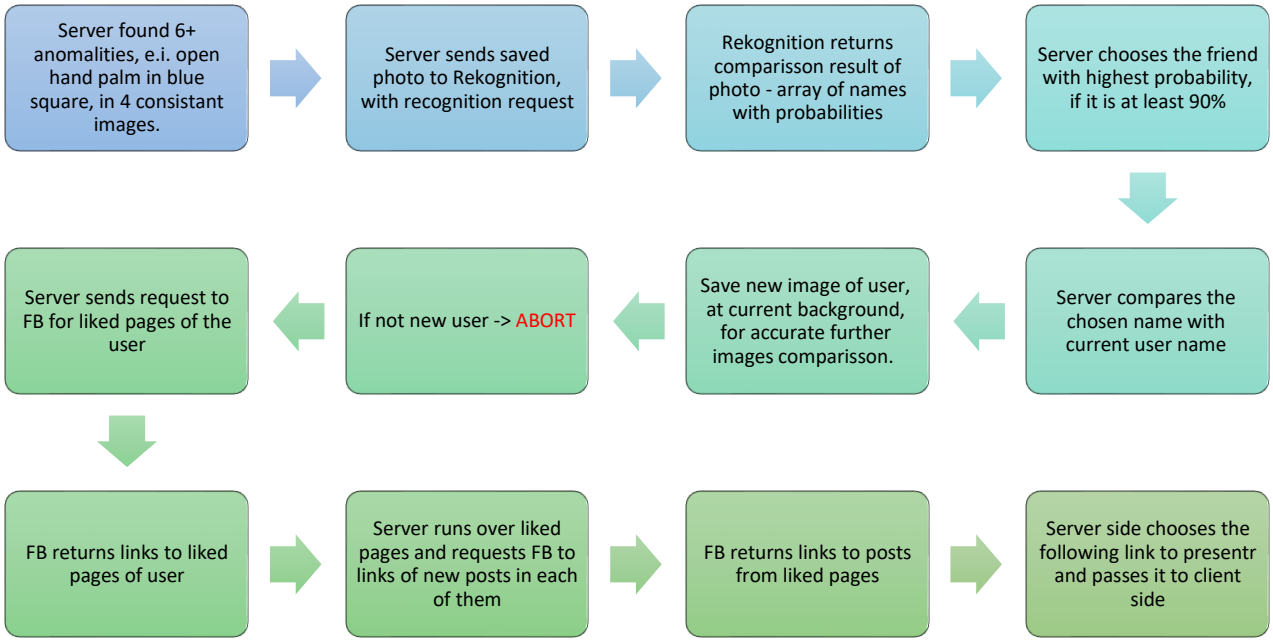


כמו כן, ברגע שמזוהה אדם חדש שהגיע, נשמור את המצב בתמונה לפני תזוזות ידיים, על מנת לדייק בהשוואות בין תמונות עוקבות.



הסבר: השרת מחלק את המלבנים הירוקים ל3 אזורים שווים בגודלם. המלבן הימני מחולק לחלקים עליון אמצעי ותחתון, והמלבן העליון מחולק לחלקים ימני, אמצעי ושמאלי.
זיהוי התנועה נעשה באמצעות שינויים בתמונה בין החלקים, לדוגמה:
בתמונה הקודמת הייתה יד בחלק הימני של המלבן העליון, בתמונה החדשה אין יותר יד בחלק זה, אך יש יד בחלק השמאלי, מכאן האלגוריתם מסיק שהייתה גלילה ימינה.

Photo Recognition Flow: Server → AWS → Server → FB



5. אכיטקטורה ומימוש

צד לקוח:

כפי שנאמר תוכנית הMAIN שלנו היינה לולאה אינסופית, אשר נקראת כל חצי שנייה, היא נראית כך:

```
var interval = 500;              // miliseconds
var time_between_posts = 1000;   // miliseconds
var first_cycles_run_counter = 0;
var ready_to_continue = true;

function run_with_timer()
{
    get_screenshot();
    if (ready_to_continue) {
        if (first_cycles_run_counter > 5) {
            set_readiness_status(true);
            get_next_post();
        } else {
            first_cycles_run_counter++; // async - takes time for
first picture to be uploaded
        }
    }
    setTimeout(run_with_timer, interval);
}

$(document).ready(run_with_timer());
```

לאחר האתחול הראשוני, אשר לוקח כמה מחזורי קריאה מכיוון שהוא אסינכרוני, הפונקצייה נכנסת למעגל של לקיחת תמונה ושליחת URL שלה לשרת.

<u>משתנים חשובים:</u>

interval – אורך המחזור שבו הסקריפט לוקח תמונה, ומעביר אותה לשרת.
time_between_posts – במידה וקיבלנו חזרה זיהוי של משתמש או תנועה, חשוב לתת לפוסט זמן להתעדכן, ולא לבצע תנועות נוספות, זהו בעצם "זמן שינה" עד שהסקריפט ממשיך לעבוד.
first_cycles_run_counter - מספר מחזורים של הסקריפט מרגע עליית העמוד עד שהוא מתחיל לשלוח בקשות לשרת, יש צורך להמתין קצת, עד שהתמונות מהמצלמה יתעדכנו בצד לקוח, עד שניתן יהיה לשלוח אותן.
ready_to_continue -זהו מנעול, הוא חוסם המשך שליחת תמונות לשרת עד שלא תתקבל תשובה כלשהי מהתמונה הקודמת.

עבור מימוש הבקשה לFLASK, נשתמש במבנה הבא עם JSON QUERY:

```
function get_next_post() {
    //document.getElementById("status_span").textContent = 'interpreting photo';
    ready_to_continue = false;

    $.getJSON(
        {
            url: "/interpretPhoto",
            data: {'image_src': document.getElementById('screenshot-img').src},
            success: function (result) {
                if (result.status != '') {
                    set_readiness_status(false);
                    if (result.status == "new_post_or_page") {
                        document.getElementById("status_span").textContent = 'caught gesture ' + result.gesture + ',
loading new post...';
                        $(".fb-post").attr('data-href', '');
                        FB.XFBML.parse();
                        $.getJSON(
                            {
                            url: "/getCurrPostUrl",
                            data: {'gesture_result': result.gesture},
                            success: function (result) {
                                document.getElementById("status_span").textContent = ''
                                document.getElementById("page_name_span").textContent = result.page_name;
                                $(".fb-post").attr('data-href', result.next_url);
                                FB.XFBML.parse();
                                set_readiness_status(true);
                                },
                            error: function (result) {
                                set_readiness_status(true);
                                }
                            });
                        }
                    else if (result.status == "new_user_gesture") {
                        document.getElementById("status_span").textContent = 'Gesture of new user caught,
identifying...';
                        document.getElementById("person_name_span").textContent = '';
                        document.getElementById("page_name_span").textContent = '';
                        $.getJSON(
                            {
                            url: "/newUserArrived",
                            data: {'photo_path': result.photo_path},
                            success: function (result_inner) {
                                if (result_inner.status == "new_person") {
                                    document.getElementById("status_span").textContent = '-';
                                    document.getElementById("person_name_span").textContent = result_inner.person_name;
                                    document.getElementById("page_name_span").textContent = result_inner.page_name;
                                    $(".fb-post").attr('data-href', result_inner.next_url);
                                    }
                                 setTimeout( function() {}, time_between_posts);
                                 FB.XFBML.parse();
                                 set_readiness_status(true);
                                 },
                            error: function (result_inner){
                                 set_readiness_status(true);
                                }
                            });
                        }
                    else {
                        window.alert('Unrecognized returned status: ' + result.status);
                        document.getElementById("status_span").textContent = '-';
                        document.getElementById("person_name_span").textContent = '-';
                        document.getElementById("page_name_span").textContent = '-';
                        FB.XFBML.parse();
                        set_readiness_status(true);
                        }
                    }
                else {
                    document.getElementById("status_span").textContent = 'waiting for orders...';
                    set_readiness_status(true);
                }
            },
            error: function (result) {
                set_readiness_status(true);
            }
        });
}
```

ניתן לראות שהפונקצייה בעצם שולחת בקשה לשרת, של "מה להציג הלאה?". הפונקצייה מעבירה בשדה הDATA לינק לתמונה האחרונה שצולמה, אותה השרת יוכל לשמור, וכאשר השרת מגיב הוא צריך להעביר סטטוס.
אם הסטטוס ריק, אזי כלום לא משתנה, ורק צריך לאפס את שדה תנועה מזוהה.

במקרים אחרים, השרת יחזיר את אחת מהתשובות:

- פוסט חדש
- משתמש חדש נקלט – במקרה כזה, הלקוח יבקש מהשרת שוב את הפוסט הבא להצגה.

בכל אחד מהמקרים הנ"ל, השרת יעביר לינק לפוסט הבא שיש להציג למשתמש, בעזרת facebook plugin.

<u>כפתורים בצד לקוח:</u>

במידה ונלחץ כפתור "new user", אז צד הלקוח יעביר לשרת בקשה עם התמונה והשם של המתשמש החדש, על מנת שהרשת ישמור אותו באוסף בAWS.
נשים לב, שלאחר אישור על הוספת המשתמש החדש, המשתמש ייתבקש להזדהות, וזאת עבור:

- לוודא שאכן הAWS מזהה אותו
- קליברציה (כיוונון) למשתמש ולרקע שמאחוריו, למען זיהוי תנועות מדויק.

```
function new_user() {
    document.getElementById("status_span").textContent = 'adding user to system'
    set_readiness_status(false);

    $.getJSON(
        {
            url: "/newUser",
            data: {'image_src': document.getElementById('screenshot-img').src,
                   'user_name': document.getElementById('new_user_input').value},
            success: function (result) {
                if (result.status != '') {
                    $(".fb-post").attr('data-href', result.next_url);
                    if (result.status == "new_user_added") {
                        window.alert('New user was added successfully, please put your left palm in
blue rectangle for calibration.')
                        document.getElementById("status_span").textContent = 'new user added to
system';
                        document.getElementById("person_name_span").textContent = result.person_name;
                        document.getElementById("page_name_span").textContent = 'Put your left palm in
blue rectangle for calibration.';
                    }
                    else {
                        window.alert('Unrecognized returned status: ' + result.status);
                        document.getElementById("status_span").textContent = '';
                        document.getElementById("person_name_span").textContent = '';
                        document.getElementById("page_name_span").textContent = '';
                    }
                    FB.XFBML.parse();
                }
                else {
                    document.getElementById("status_span").textContent = '';
                }
                set_readiness_status(true);
            },
            error: function (result) {
                window.alert('Error in saving new user: ' + result);
                set_readiness_status(true);
            }
        });
}
```

אם נלחץ על כפתור ניקוי האוסף של המשתמשים, גם כאן תועבר בקשה לשרת, ונחכה לאישור:

```
function clear_col() {
    document.getElementById("status_span").textContent = 'clearing users'
    set_readiness_status(false);
    $.getJSON(
        {
            url: "/clrCol",
            data: {},
            success: function (result) {
                if (result.status == 'collection cleared') {
                    window.alert(result.status);
                } else {
                    window.alert('error in clearing');
                }
                setTimeout( function() {}, time_between_posts);
                set_readiness_status(true);
                document.getElementById("status_span").textContent = ''
            },
            error: function (result) {
                window.alert('Error in clearing users from collection: ' + result);
                set_readiness_status(true);
                document.getElementById("status_span").textContent = ''
            }
        });
}
```

כפתור אחרון הינו כפתור הסביבה החדשה, הכפתור נועד לקלוט את הסביבה שהמצלמה רואה, ומול תמונה זו ניתן להשוות תמונות חדשות, כדי לבדוק אם יש סימן בריבוע הכחול על הגעת משתמש. בלחיצה תשלח בקשה לשרת לשמור את התמונה:

```
function save_new_env() {
    document.getElementById("status_span").textContent = 'saving image of environment'
    set_readiness_status(false);

    $.getJSON(
        {
            url: "/newEnvironment",
            data: {'image_src': document.getElementById('screenshot-img').src},
            success: function (result) {
                if (result.status == 'new_env_saved') {
                    window.alert('New environment saved successfully.');
                }
                else {
                    window.alert('Unrecognized return value: ' + result.status);
                }
                setTimeout( function() {}, time_between_posts);
                set_readiness_status(true);
            },
            error: function (result) {
                window.alert('Error in saving new environment: ' + result);
                set_readiness_status(true);
            }
        });
}
```

פונקציות נוספות:

פונקציית הזרמת תמונה ללא הפסקה (STREAMING):

```
navigator.mediaDevices.getUserMedia(constraints).then(handleSuccess).catch(handleError);
        function handleSuccess(stream) {
            video.srcObject = stream;
        }
        function handleError(error) {
            console.error('Error: ', error);
        }
```

פונקציית לקיחת SNAPSHOT רגעי מהמצלמה:

```
        function get_screenshot()
        {
            canvas.width = video.videoWidth;
            canvas.height = video.videoHeight;
            canvas.getContext('2d').drawImage(video, 0, 0);
            img.src = canvas.toDataURL('image/jpeg');
        }
```

צד שרת:

צד שרת נכתב בפייתון, בגלל נוחות התממשקות עם ספרייה קיימת FBRECOG אשר כתובה בפייתון (תזכורת – זוהי ספרייה לזיהוי פנים בתמונה ע"י פייסבוק), וכמו כן שימוש בFLASK, שזהו הAPI לתקשורת עם צד לקוח.
הערה חשובה: כפי שהוסבר בפרק "אתגרים בפיתוח" לבסוף פתרון FB-RECOG לא נלקח משום שפייסבוק חסמו אפשרות להעלאת תמונה, אך עבור פרוייקט זה הדבר לא היווה בעיה.
שירות הRekognition של AWS גם הוא עובד עם פייתון, וההתמשקות איתו הייתה מהירה וקלה.

השרת מכיל פונקציית תגובה לבקשה של צד לקוח לזיהוי תמונה:

```
@app.route("/interpretPhoto", methods=['GET'])
def interpret_photo():
    global sequel_images_counter

    path = request.args.get('image_src')  # image link passed from javascript

    sequel_images_counter = (sequel_images_counter % SEQUEL_PHOTOS_TO_KEEP) + 1 # we want 1 and 2
    next_photo_name = PHOTO_NAME_PATTERN.format(sequel_images_counter)
    if sys.version_info[0] <= 2:
        import urllib
        urllib.urlretrieve(path, next_photo_name)
    elif sys.version_info[0] <= 3:
        import urllib.request
        urllib.request.urlretrieve(path, next_photo_name)

    res_dict = _interpret_photos(next_photo_name)

    return jsonify(res_dict)
```

ונשים לב שניתן להעביר מידע מהלקוח לשרת בצורה הבאה:

```
path = request.args.get('image_src') # image link passed from javascript
```

זוני בעצם הפונקציה הראשית של צד שרת. פונקציה זו אחראית על השוואת תמונות עוקבות לצורך זיהוי תמונות ידיים, ושליחת בקשה לפייסוק לזיהוי פנים בעזרת פונקציית עזר: `_get_person_name`, אם עומד מול המצלמה משתמש חדש, היא תפנה לאלגוריתם זיהוי עמודים מועדפים (יוסברבהמשך) .
הפונקציה תחזיר לצד לקוח עדכון עם הפוסט הבא להצגה + מידע על המשתמש והעמוד.

פונקציות עזר:

```
def _new_person_retrieve_data(username)
def _try_to_recognize(image_path)
def _interpret_photos(last_photo_path)
def _create_liked_posts_list(index_to_create_for)
def _create_liked_pages_list()
def _is_new_user_gesture(img_name_new)
def _fill_hand_matrices(image)
def _get_hand_gesture(img_name_new)
def _compare_images_for_gesture(previous_hands_matrix_right, new_hand_matrix_right,
                                previous_hands_matrix_top, new_hand_matrix_top)
def _login(session, email, password):
```

כל הפונקציות אחראיות על התממשקות עם AWS, זיהוי תמונות, זיהוי תנועוץ וכמובן התחברות _login כאשר רוצים לייבא פוסט.

בנוסף קיימת כמובן פונקציית MAIN שמריצה את שרת הFLASK הוירטואלי:

```
if __name__ == '__main__':
    app.run(debug=True)
```

<u>זיהוי בן אדם:</u>

על מנת לזהות בן אדם, עלינו קודם כל ליצור עבורו וקטורי של תווי פנים בשירות Rekognition. הדבר מתבצע כאשר נקראת הפונקצייה הבאה מצד לקוח:

```
@app.route("/newUser", methods=['GET'])
def new_user():

    path = request.args.get('image_src')  # image link passed from javascript
    username = request.args.get('user_name')  # new user name from javascript
    if sys.version_info[0] <= 2:
        import urllib
        urllib.urlretrieve(path, PHOTO_USER_ADDED)
    elif sys.version_info[0] <= 3:
        import urllib.request
        urllib.request.urlretrieve(path, PHOTO_USER_ADDED)

    try:
        client = boto3.client('rekognition')
        colId = "SmartSocNet"
        #c = client.create_collection(CollectionId=colId)
        with open(PHOTO_USER_ADDED, "rb") as imgf:
            img = imgf.read()
        indr = client.index_faces(CollectionId=colId, Image={'Bytes': img},
ExternalImageId=username, MaxFaces=1, )
    except Exception as err:
        print(err.msg)
        raise

    res_dict = {'status': 'new_user_added',
                'person_name': 'username'}

    return jsonify(res_dict)
```

נזכור שהפונקציה נקראת בלחיצה על כפתור "new user" , וצד הלקוח מעביר לשרת שם שהוזן ותמונה עדכנית. פונקצייה זו תכניס את האדם לאוסף: `colId = "SmartSocNet"`

ע"י הפקודה: `indr = client.index_faces(CollectionId=colId, Image={'Bytes': img}`

פונקצייה הנקראת בבקשת ניקוי האוסף בשרתי AWS:

```
@app.route("/clrCol", methods=['GET'])
def clear_collection():
    res_dict = {'status': ''}
    client = boto3.client('rekognition')
    colId = "SmartSocNet"
    client.delete_collection(CollectionId=colId)
    c = client.create_collection(CollectionId=colId)

    res_dict['status'] = 'collection cleared'
    return jsonify(res_dict)
```

פונקציית שמירה של סביבה חדשה:

```
@app.route("/newEnvironment", methods=['GET'])
def new_environment():

    path = request.args.get('image_src')  # image link passed from javascript
    if sys.version_info[0] <= 2:
        import urllib
        urllib.urlretrieve(path, PHOTO_ENVIRONMENT)
    elif sys.version_info[0] <= 3:
        import urllib.request
        urllib.request.urlretrieve(path, PHOTO_ENVIRONMENT)

    res_dict = {'status': 'new_env_saved'}
    return jsonify(res_dict)
```

פונקציית בקשה של הלקוח, לקבל לינק עבור משתמש חדש שהגיע (תזכורת – הלקוח יודע שהגיע משתמש חדש, כיוון שהשרת נתן איתות על כך):

```
@app.route("/newUserArrived", methods=['GET'])
def new_user_arrived():
    global sequel_photos_with_new_user_gesture

    photo_path = request.args.get('photo_path')
    res_dict = _try_to_recognize(photo_path)
    sequel_photos_with_new_user_gesture = 0
    return jsonify(res_dict)
```

פונקצייה אשר מחזירה לינק לפוסט הבא להצגה לצד לקוח, תזכורת – השרת מאותת ללקוח שהייתה תנועה, וכעת הלקוח מעוניין לקבל את הלינק המתאים עבורה:

```
@app.route("/getCurrPostUrl", methods=['GET'])
def get_curr_post_url():
    global liked_page_index
    global liked_post_index
    global pages_to_show  # list of dicts

    res_dict = {'status': ''}
    gesture_result = request.args.get('gesture_result')
    if gesture_result == 'scroll_up':
        if 'posts' not in pages_to_show[liked_page_index]:
            _create_liked_posts_list(liked_page_index)
        liked_post_index = (liked_post_index + 1) % len(pages_to_show[liked_page_index]['posts'])
        res_dict['status'] = 'new_post'
    if gesture_result == 'scroll_down':
        if 'posts' not in pages_to_show[liked_page_index]:
            _create_liked_posts_list(liked_page_index)
        liked_post_index = (liked_post_index - 1) % len(pages_to_show[liked_page_index]['posts'])
        res_dict['status'] = 'new_post'
```

```
    if gesture_result == 'swipe_left':
        liked_page_index = (liked_page_index + 1) % len(pages_to_show)
        liked_post_index = 0
        res_dict['status'] = 'new_page'
    if gesture_result == 'swipe_right':
        liked_page_index = (liked_page_index - 1) % len(pages_to_show)
        liked_post_index = 0
        res_dict['status'] = 'new_page'

    res_dict['page_name'] = pages_to_show[liked_page_index]['name']
    if len(pages_to_show) > liked_page_index:
        res_dict['gesture'] = gesture_result
        if len(pages_to_show) > 0:
            if 'posts' not in pages_to_show[liked_page_index]:
                _create_liked_posts_list(liked_page_index)
            res_dict['next_url'] = pages_to_show[liked_page_index]['posts'][liked_post_index]

    return jsonify(res_dict)
```

חשוב לשים לב!

באפשרות החינמית של שימוש בrekognition, אנו מקבלים רק 5000 זיהויים חינמיים בחודש. תחילה המערכת שלנו העלתה כל תמונה רביעית לצורך זיהוי, אך מנגנון זה הוחלף בהעלאת רק תמונות "חשודות" להגעת משתמש חדש.

נזכיר שעל מנך שהמערכת תחפש אדם חדש, עליו לעשות "שלום" בריבוע הכחול.

<u>זיהוי עמודים מועדפים והעברתם לשרת:</u>

זיהוי עמודים מועדפים של החבר יתבצע בעזרת הפונקצייה: `_create_liked_pages_list()`. הפונקצייה מכירה את האדם שכרגע מול האפליקציה (הוא כבר זוהה), ולכן היא יכולה להתחבר כמשתמש הראשי לפייסבוק, ולהגיע דרך URL לעמודים הלייקים של המשתמש הנוכחי (בעזרת `_login`):

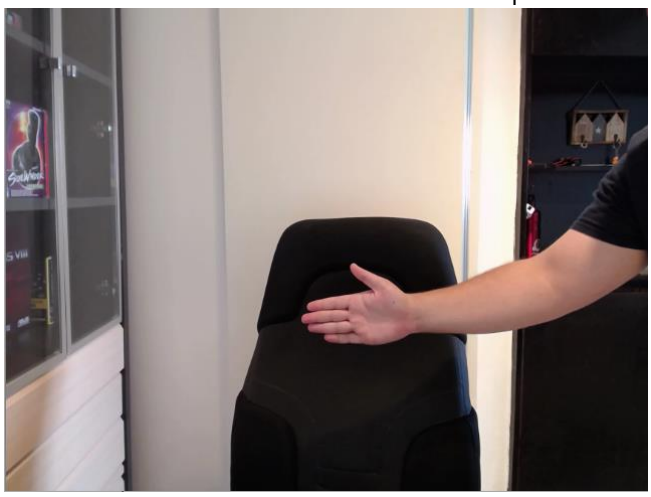
```
likes_url = 'https://www.facebook.com/' + dotted_name_of_curr_friend
+'/likes?lst=1198688678%3A843054236%3A1535896155'
response = session.get(likes_url, cookies=cookies, allow_redirects=False)
```

מכאן נשתמש בספרייה של BEAUTIFUSOUP שעליה הסברנו קודם, על מנת להוציא מעמוד הלייקים את השמות והלינקים של העמודים והפוסטים האהובים, ולבסוף לשלוח לצד לקוח את הפוסט הבא להצגה.

<u>זיהוי תנועות ידיים:</u>
עלינו לזהות 4 תנועות ידיים: למעלה, למטה, משיכה לימין ומשיכה לשמאל.

פתרון ראשון שמומש, אך לבסוף הוחלט לוותר עליו עבד כך:
פונקציה קיבלה 2 תמונות, ולאחר הפעלת האלגוריתם המוסבר להלן על כל אחת מהן, שלחה את התוצאה ללקוח. האלגוריתם:

- תמונה מקורית:



- הפוך את התמונה לגווני שחור – לבן:

- טשטש את התמונה מעט על מנת להמנע מרעשים:

- קיטוב צבעי התמונה לשחור אם הצבע בין 0 ל100, וללבן אם הצבע בין 101-255:

- וכעת היה ניתן לחשב את ממוצע הצבע הלבן בתמונה, למשל: [285, 261].

- בשלב זה היו לנו 2 ממוצעים של צבעים לבנים ב2 תמונות. נזכור שהתמונות הן במרווח זמן של חצי שנייה. יכולנו לראות לאיזה כיוון זז ממוצע הצבע הלבן בתמונה השנייה. לפי כיוון התזוזה המקסימלית חישבנו אם תנועת היד הייתה בכיוון מעלה \ מטה \ ימינה \ שמאלה.

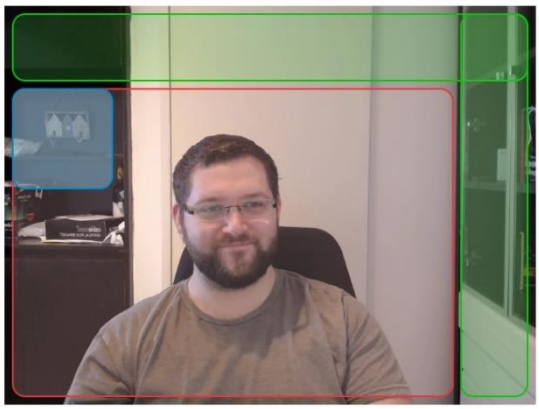
כמו שצוין האלגוריתם לא עבד, הסיבות לכך היו נעוצות בכמה גורמים:

- הצבת מצלמה מול רקע בהיר, הייתה מתעתעת את השחור והלבן בתמונה, היה צורך לתת ידנית את גבול הקיטוב ללבן ושחור עבור כל סביבה חדשה.
- אפילו בהנתן רקע אחיד וכהה, עדיין שינויים בתאורה בין יום ולילה היו דורשים קליברציה מחודשת של הקיטוב.
- לאנשים שונים צבעי עור שונים, וגם צבעי חולצה שונים, ייתכן שהתנועה פשוט לא הייתה נקלטת.
- כתלות אם האדם עומד רחוק או קרוב למצלמה, תזוזת הצבע הלבן בין 2 תמונות משתנה בעוצמתה, כך שלא ניתן היה לדעת אם האדם זז מעט, או ממש הזיז את היד.
- כדי לעשות תנועה מול המצלמה המשתמש חייב לביא את ידו למסך, אך האלגוריתם עלול לחשוב בטעות שהייתה גלילה מטה, מכיוון שיד עלתה מלמטה למעלה!
- בהמשך לבעיה הקודמת, כאשר המשתמש מזיז את היד מלמעלה למטה, המרפק שלו נכנס לתמונה ו"מבלבל" את הצבע הלבן.

<u>הפתרון שהתקבל:</u>

העבודה על דיוק קליטת התנועה, התבצעה בכמה מישורים:

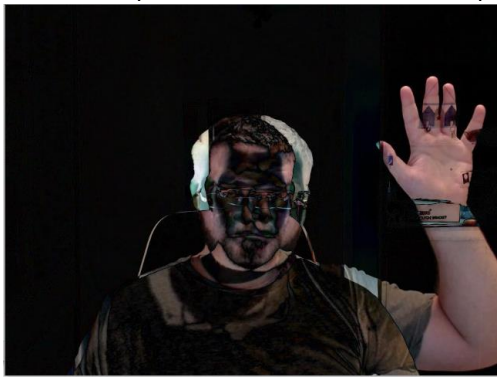
- ראשית כמו שהוצג בפרק "מבט על", המסך חולק לכמה חלקים.



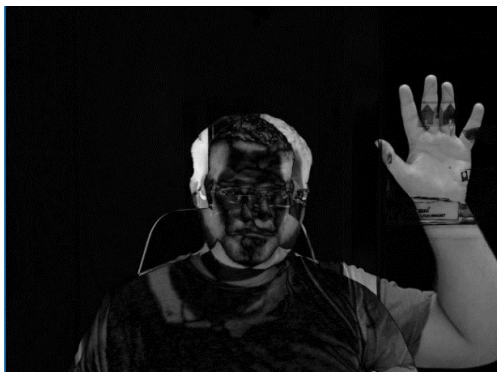
על מנת להודיע למערכת שהגיע אדם חדש, עליו לעשות שלום עם כף יד פתוחה בריבוע הכחול. רק שם, המערכת תנסה לזהות אותו.
כמו כן המערכת מבקשת מהאדם לעמוד במרכז המלבן האדום.
הבקשה לא מגיעה באופן מקרי, כאשר אדם יעמוד במרכז המבלן האדום, כך שכף ידו כמעט ממלאה את הריבוע הכחול, הוא יהיה במרחק האופטימלי עבור המערכת לקלוט תנועות במלבנים הירוקים.
כמו כן, במצב כזה, כאשר רוצים לדפדף למעלה ולמטה, יד המשתמש תגיע למלבן הירוק הימני מהצד בלבד, כך שלא יהיה בלבול של המערכת כאשר המרפק בחלק הימני התחתון.

- צעד נוסף לעבודה מדויקת היה מעבר מתמונת שחור-לבן, לתמונת שינויים, כלומר במקום לקחת 2 תמונות, לבצע עליהן אלגוריתם של קיטוב שחור ולבן ואחכ למדוד את שינוי מרכז הכובד של הצבע הלבן, נעשה אלגוריתם אחר:

1. קח 2 תמונות ותעשה עליהן DIFF לפי פיקסלים, מתקבלת תמונה של שינויים בלבד):



2. כעת נעביר אותה לגווני אפור:



3. נקטב את התמונה לשחור ולבן אבסולוטיים:

4. נטשטש את התמונה כדי להמנע מרעשים:

5. נקטב שוב, על מנת להשאר עם קווים ברורים:

6. מהתמונה האחרונה ניתן לחתוך את החלקים שמעניינים אותנו בהתאם למיקום המלבנים (כחול לזיהוי וירוק לתנועה).

- הצעד האחרון שבוצע למען דיוק בתמונה, היא לקיחת תמונות רקע. הפתרון עזר מכיוון שאנחנו בוחנים כעת DIFF בין תמונות, וכדי לתת דיוק לDIFF, צריך לשמור תמונה ללא סימני ידיים.

1. כשמים את המצלמה במקום חדש כדאי לשמור את הרקע עם כפתור NEW ENVIRONMENT, כדי שבקלות ניתן יהיה לזהות תנועות ידיים של "שלום" כשהגיע משתמש חדש.
2. כשזוהה משתמש חדש, נשמור את התמונה שלו כאשר הוא עשה "שלום" למצלמה, מכיוון שבדרך זו מתקבלים מלבנים ירוקים נקיים מתנועה, עם התאורה באותו רגע של שימוש.

6. סיכום

ניתן לראות שהתממשקות עם פייסבוק וAWS הינן אפשריות, עם עמידה באתגרי אבטחה. זיהוי פנים ותנועות ידיים מבוצע בצורה טובה ומהירה עם כל מצלמה.
המסקנה היא שניתן ליצור בית חכם ונוח לשימוש כפי שהוגדר ביעדים.

7. Requirements

1. Manual assumes you already have user in AWS and Rekognition service.
2. Install flask + activate virtual machine from here:
   http://flask.pocoo.org/docs/1.0/installation/
   Run it.
3. Make sure you have python 2.7 and up on the virtual machine.
4. With pip install in virtual environment:
   - Install beautifulsoup (for FB html parsing)
   - Install boto3 (AWS – Rekognition API)
5. Install OpenCV2 with following steps:
   - On windows:
     From virtual environment:
     pip install opencv-python
   - On RaspberryPi3 (Raspbian):
     There is no install file for RP3, therefore it is required to download opencv libraries, and then compile and install them, the process takes approximately 2-3 hours. Follow the instructions from here:
     https://www.pyimagesearch.com/2016/04/18/install-guide-raspberry-pi-3-raspbian-jessie-opencv-3/
6. Enter the following environment parameters to virtual machine with your values:
   - export AWS_DEFAULT_REGION=us-east-1
   - export AWS_ACCESS_KEY_ID=<your_value>
   - export AWS_SECRET_ACCESS_KEY=<your_value>
   - export USER_MAIL_PROJ=<your_value>
   - export PASSWORD_PROJ=<your_value>
7. Run Facebook in your browser with your user.
8. Run the program (flask run).

In case of security error with AWS, run following command (with correct date and time) and wait a minute:
sudo timedatectl set-time '2018-11-05 15:58'
It is strongly advised to use Ethernet cable and not WiFi.

In case you want to run with raspberry camera:
sudo modprobe bcm2835-v4l2
But note it might not work with Javascript.

8. נספחים

קישור לפרוייקט בGitHub:

https://github.com/alexey-khromov/social-network-smart-house

JSON:

מתוך ויקיפדיה: https://he.wikipedia.org/wiki/JSON

JSON ראשי תיבות של (**J**ava**S**cript **O**bject **N**otation) הוא פורמט טקסטואלי, הקריא לאדם, המיועד להעברת מבני מידע המורכבים מזוגות של מפתח-ערך. השימוש העיקרי של הפורמט הוא להעברת מידע בין שרת לצרכן כתחליף לפורמט XML.